ProjectPlace API Documentation

OAuth2.0

OAuth 2.0 is a protocol that lets external applications request authorization to private details in a user's ProjectPlace account without getting their password.

You can find a client library to simplify your implementation here.

ProjectPlace supports two OAuth2 flows as documented below:

- Authorization code flow - for standard user authentication of your application
- Client credentials flow - for application to application communication - only relevant for account wide integrations - see Enterprise Integrations

Authorization code flow

The Authorization code flow is the standard way in which you get authorization from an end user to access and manipulate their data.

Note - The redirection endpoint requires the use of TLS(HTTPS) as the redirection request will result in the transmission of sensitive credentials over an open network.

Note - All POST requests below must be sent with `Content-Type: application/x-www-form-urlencoded`

Step 1. Redirect users to request ProjectPlace access

The first step is to direct the user to the following endpoint, where he/she will be promted to authorize your application.

```
GET https://api.projectplace.com/oauth2/authorize
```

Parameters	Description
`client_id`	**Required**. The client ID received from ProjectPlace when you registered. This is the same as the `application key` found in your application management.
`state`	**Recommended**. An unguessable random string. It is used to protect against cross-site request forgery attacks.
`redirect_uri`	The URL in your app where users will be sent after authorization. This MUST be the same as given in your application management.

If the user authorizes your application, the client will redirect to the `redirect_uri` with an authorization code appended to the URL. For example:

```
/redirect_uri&code=hdb764jhl1hg345hgdbn
```

If the user denies access an error code is instead appended, such as:

```
/redirect_uri&error=access_denied
```

Note The redirected uri will also included state parameter if provided in the request. You must validate this state parameter so that it is the same as originally sent before accepting the authorization code for use.

Step 2. Exchange code for access_token

To get access to the user's resources you must exchange your authorization code for an access token using the following endpoint

```
POST https://api.projectplace.com/oauth2/access_token
```

Parameters	Description
`client_id`	**Required**. The client ID (application key) of your application.
`client_secret`	**Required**. The client secret received from ProjectPlace when you registered.
`code`	**Required**. The authorization code given after step 1.
`grant_type`	**Required**. Should be `"authorization_code"`

Example response:

```
{
    token_type: "Bearer",
    access_token: "ed6f21f9bda1d7feb00999cd9b2eab94",
    expires: 86400,
    refresh_token: "2fe617659315326ce019"
}
```

Where `Bearer` simply means that the access token can be used as is without any extra encoding/decoding. The `access_token` is valid for 30 days, after which you will have to request a new access token.

You use the access token by including it in the Authorization headers of your API requests such as `Authorization: Bearer {YOUR_ACCESS_TOKEN_HERE}`

A new access token can be requested without involving the end user thanks to the refresh token (which is valid for 120 days).

Step 3 - Refresh access token

As long as you have a valid access token, you can request resources on behalf of the user. But since the access token is only valid for 30 days you will need to refresh the access token, using the refresh token.

This is how you refresh your access tokens.

```
POST https://api.projectplace.com/oauth2/access_token
```

Parameters	Description
`client_id`	**Required**. The client ID (application key) of your application.
`client_secret`	**Required**. The client secret received from ProjectPlace when you registered.
`refresh_token`	**Required**. Refresh token given with access_token.

`grant_type`	**Required**. Should be "`refresh_token`"

Example response:

```
{
    token_type: "Bearer",
    access_token: "fd6f21f9bda1d7feb00999cd9b2acb94",
    expires: 86400,
    refresh_token: "fac334659345326cefac"
}
```

The response looks exactly like the one given by step 2. As you can see you receive a new access token, as well as a new refresh token. The old access token and refresh token are no longer valid, and you will have to use the new refresh token to repeat the process.

This basically means that as long as you use the refresh token once every 120 days you can maintain access to the user's resources without asking the user to re-authorize your application.

Step 4. Access APIs

To access resources on behalf of the user you have two options, you can either supply an `Authorization` header like this:

```
Authorization: Bearer ed6f21f9bda1d7feb00999cd9b2eab94
```

Or you can add it to your query strings when accessing API endpoints such as:

```
https://api.projectplace.com/1/user/me/projects?access_token=ed6f21f9bda1d7feb00999cd9b
```

Client Credentials Flow

The client credentials flow is only relevant for "robot" accounts. Using the client credentials flow you can get an access token by simply supplying `client_id` and `client_secret` directly to the access token endpoint

```
POST https://api.projectplace.com/oauth2/access_token
```

Parameters	Description
`client_id`	**Optional**. The client ID (application key) of your application.
`client_secret`	**Optional**. The client secret received from ProjectPlace when you registered.
`grant_type`	**Required**. And must be `client_credentials`

The `client_id` and `client_secret` are optional. However, if they are omitted they MUST instead be sent encoded in a an Authorization header - using the Basic HTTP Authentication Scheme.

A successful request renders an access token - but no refresh token. In an application to application setting there is no need for a refresh token. To get a new access token - simply repeat the client credentials flow.